RECEIVED
CENTRAL FAX CENTER

FEB 1 5 2007

Amendments to the Claims

The following listing of claims supersedes all previously pending claims:

1. (Currently amended): A computer implemented method for data flow control for ~~of~~ a plurality of execution nodes of an adaptive computing engine (ACE), the method comprising:

(a) associating a plurality of task parameters with a plurality of tasks within ~~an~~ each of the plurality of execution nodes;

(b) identifying readiness of ~~a~~ the plurality of tasks ~~resources~~ based on a status of the plurality of task parameters; and

(c) pacing allocation of the plurality of tasks to the plurality of execution nodes based on the readiness of the plurality of tasks ~~resources~~.

2. (Currently amended): The computer implemented method of claim 1 wherein the execution node includes a reconfigurable execution unit.

3. (Currently amended): The computer implemented method of claim 2 wherein the reconfigurable execution unit further comprises a one or more finite state machines.

4. (Currently amended): The computer implemented method of claim 1 wherein the task parameters identify by designation, an input port, an output port, a finite state machine, and a finite state machine instance.

5. (Currently amended): The computer implemented method of claim 4 wherein the identifying readiness step (b) further comprises ~~the step of (b1)~~ identifying a task as an executable task when the input port is available, the output important is available, and the finite state machine is idle.

6. (Currently amended): The computer implemented method of claim 5 ~~claim 1~~ further comprising after the pacing allocation ~~the step of (d)~~ aggregating executable tasks in a queue.

7. (Currently amended): The computer implemented method of claim 6 wherein ~~allocation~~ the pacing allocation step (c) further comprises: ~~the steps of (c1)~~

reading a next executable task from the queue, and ~~(c2)~~

generating a signal to start execution in ~~the~~ a finite state machine associated with the next executable task.

8. (Currently amended): The computer implemented method of claim 7 further comprising: ~~the steps (e) of~~

after the aggregating, reconfiguring the finite state machine from one instance to another as necessary;

reading data from the input port; ~~(f)~~

processing the data in the finite state machine; and (g)

writing data to the output port.

9. (Currently amended): The computer implemented method of claim 8 further comprising: ~~the steps of (h)~~

generating a signal indicating completion of ~~the~~ execution in the finite state machine; and

~~(c)~~

re-entering an idle state by ~~in~~ the finite state machine.

10. (Currently amended): The computer implemented method of claim 4 wherein the designation comprises a number.

11. (Currently amended): A system for data flow control in processing nodes of an adaptive computing engine (ACE), the system comprising:

a reconfigurable execution unit; and

flow control logic coupled to the reconfigurable execution unit, ~~for~~ the flow control logic configured for associating tasks and task parameters, identifying readiness of the tasks ~~resources~~ based on a status of the task parameters, and pacing allocation of the tasks to the reconfigurable execution unit based on the readiness of tasks ~~resources~~.

12. (Original): The system of claim 11 wherein the reconfigurable execution unit further comprises a one or more finite state machines.

13. (Currently amended): The system of claim 11 wherein the task parameters identify[,] by designation, an input port, a finite state machine, and a finite state machine instance.

14. (Original): The system of claim 13 wherein the designation comprises a number.

15. (Currently amended): The system of claim 13 wherein the flow control logic further configured to identify ~~identifies~~ a task as an executable task when ~~the~~ an input port is available, ~~the~~ an output port is available, and the finite state machine is idle.

16. (Currently amended): The system of claim ~~12~~ 15 further comprising a queue for aggregating a plurality of executable tasks.

17. (Currently amended): The system of claim 16 wherein the flow control logic is further configured to read ~~reads~~ a next executable task from the queue and to generate ~~generates~~ a signal to start execution in the finite state machine associated with ~~the~~ a next executable task.

18. (Currently amended): The system of claim 13 wherein the finite state machine is configured to reconfigure ~~reconfigures~~ from one instance to another, and, if necessary, to read ~~reads~~ data from the input port, to process ~~processes~~ the data, and to write ~~writes~~ the data to the output port.

19. (Original): The system of claim 18 wherein the finite state machine is further configured to generate ~~generates~~ a signal indicating completion of the execution and to re-enter ~~re-enters~~ an idle state.

20. (Currently amended): A system for flow control in processing nodes of an adaptive computing engine (ACE), the system comprising:

a plurality of finite state machines, wherein each of the plurality of finite state machines are configured to perform a ~~plurality of executable tasks~~ task;

control logic for determining task parameter status for the task and for identifying the task as executable; and

a task queue for storing a plurality of executable tasks transferred by the control logic and for issuing the plurality of executable task to the plurality of finite state machines.

21. (Currently amended): The system of claim 20 wherein the plurality of finite state machines form an execution unit for a processing node within ~~an~~ the adaptive computing engine.

22. (Currently amended): The system of claim 20 wherein the control logic determines a status of an input port, ~~and~~ an output port, a finite state machine ~~idle state~~, and an instance of the finite state machine.

23. (Currently amended): The system of claim 22 wherein the control logic identifies a task as executable when the input port and the output port are available and the finite state machine is idle.

24. (new): A computer implemented method for controlling data flow of a plurality of heterogeneous nodes on an adaptive computing engine (ACE), the plurality of heterogeneous nodes including a plurality of reconfigurable finite state machines, the method comprising:

constructing an active task list, the active task list including a plurality of tasks for execution on one of the plurality of finite state machines, the constructing including,

determining a status for each of the plurality of tasks, wherein the status is determined from a plurality of task parameters, and

pacing each of the plurality of tasks in the active task list when the status indicates readiness; and

executing each of the plurality of tasks in the active task list on one of the plurality of reconfigurable finite state machines, wherein each of the plurality of reconfigurable finite state machines are dynamically reconfigurable to execute each of the plurality of tasks when required.

25. (new): The computer implemented method of claim 24 further comprising:

generating a completion signal when each of the plurality of tasks is completed on each of the plurality of finite state machines; and

returning each of the plurality of finite state machines to an idle condition.

26. (new): The method of claim 25 wherein the plurality of task parameters identify by designation an input port, an output port, a finite state machine, and a finite state machine instance.

27. (new): The method of claim 24 wherein functionality of the plurality of reconfigurable finite state machines is selected from the group consisting of: FIR filtering, IIR filtering, FFT, DCT, IDCT, convolution/correlation, convolutional encoding, Viterbi decoding, Huffman encoding, Huffman decoding, encryption and decryption, and LSFR/PN sequence generation.